SOFTWARE ENGINEERING LABORATORY SERIES

SEL-84-003

INVESTIGATION OF SPECIFICATION MEASURES FOR THE SOFTWARE ENGINEERING LABORATORY (SEL)

DECEMBER 1984

(NASA-TM-88591) INVESTIGATION OF SPECIFICATION MEASURES FOR THE SOFTWARE ENGINEERING LABORATORY (SEL) (NASA) 74 p
HC A04/MF A01 CSCL 09B
N86-19983
Unclas
G3/61 05494

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

SOFTWARE ENGINEERING LABORATORY SERIES SEL-84-003

INVESTIGATION OF SPECIFICATION MEASURES FOR THE SOFTWARE ENGINEERING LABORATORY (SEL)

DECEMBER 1984

NASA

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organizational members:

- NASA/GSFC (Systems Development and Analysis Branch)
- The University of Maryland (Computer Sciences Department)
- Computer Sciences Corporation (Flight Systems Operation)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/TM-84/6162.

The primary contributors to this document are

William Agresti	(Computer Sciences Corporation)
Victor Church	(Computer Sciences Corporation)
Frank McGarry	(Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 552
NASA/GSFC
Greenbelt, Maryland 20771

ABSTRACT

Requirements specification measures are investigated for potential application in the Software Engineering Laboratory. Eighty-seven candidate measures are defined; sixteen are recommended for use. Most measures are derived from a new representation, the Composite Specification Model, which is introduced in this document. The results of extracting the specification measures from the requirements of a real system are described.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

SECTION 1 - INTRODUCTION

The purpose of this document is to report on an investigation of requirements specification measures for possible use in the Software Engineering Laboratory (SEL). This section presents the study objective, analytical approach, and document organization.

1.1 OBJECTIVE OF THE STUDY

The objective of the study was to investigate measures that provide a quantitative characterization of the size and nature of the software requirements. The measures will be useful to managers for assessment and prediction and will provide answers to the following questions:

- Is the intended behavior of the system understood?
- How large will the system be?
- Will the system be especially expensive to develop?

The difficulty with meeting the objective lay in the timing of the measurement. To be useful, the measures must depend on information that is available during the requirements analysis phase of the software development life cycle. At the beginning of this phase, the functional specifications and requirements document (FSRD) should be complete (Reference 1). At the end of the phase, preliminary design begins. The difficulty is that most familiar measures, like lines of code, are not known until later in the life cycle. During requirements analysis, the system is represented by documents like the FSRD, which do not facilitate objective measurement.

1.2 ANALYTICAL APPROACH

The analytical approach was strongly influenced by the need for the measures to be based on information potentially

available during the requirements analysis phase. Most previous approaches to specification measures have experimented with subjective evaluations. By designing questionnaires and scoring the responses, similar subjective measures could be developed for the flight dynamics environment. The subjective nature of such measures, however, makes them less desirable than objective measures in forming the foundation of a program of assessment and prediction. For this reason, objective measures have been pursued in this study.

The identification of specific items to measure was driven by the planned use of the measures. The basic needs are to assess the size and nature of the requirements and to predict the effort required to develop software that satisfies the requirements. One attempt was made to identify requirements characteristics that lead to errors so that those characteristics could be included among the items being measured. The brief study was inconclusive, however, as summarized in the appendix.

From the consideration of information availability, a three-step approach emerged: First, define measures derived from documentation routinely prepared during requirements analysis. Second, identify additional information that is available during requirements analysis but not currently captured in documentation. Define a second set of measures based on this additional information. Third, extract the measures from the requirements for a real system.

1.3 DOCUMENT ORGANIZATION

The organization of the document reflects the three-step approach. Section 2 defines measures based on existing documentation, principally the FSRD. Section 3 introduces a new representation, the Composite Specification Model (CSM), which captures additional requirements information. The

measures defined in Section 4 are derived from the CSM representation. Section 5 presents the results of an exercise to extract the measures, and Section 6 presents an assessment of the specification measures investigation.

SECTION 2 - MEASURES BASED ON EXISTING DOCUMENTATION: M1 TO M29

This section introduces possible measures M1 through M29. These measures are not necessarily recommended for use; they are presented as part of a survey of possible measures. The measures were obtained through consultation with colleagues and a review of related literature. The measures are defined in Section 2.1, and a preliminary assessment of their utility, costs, and benefits is presented in Section 2.2.

All of the measures discussed in this section are based on information that is captured routinely as part of the software development process, the FSRD being the principal source. Table 2-1 lists these measures.

2.1 DEFINITION OF MEASURES M1 TO M29

2.1.1 Basic Volume Measures

Measures M1 through M7 can be obtained from an examination of the FSRD.

- M1 Number of Pages in the FSRD
- M2 Number of Paragraphs in the FSRD (see Reference 2, p. 482)
- M3 Number of Instances of the word "shall" in the FSRD (see Reference 2, p. 482)
- M4 Number of Figures in the FSRD
- M5 Number of Tables in the FSRD
- M6 Number of Equations in the FSRD
- M7 Number of Variables in the FSRD

Although they are easy to obtain, these measures are unreliable. They are simple counts that would have value only if there is some discipline in the way the FSRD is prepared: same scope of contents, same level of detail in the descriptions, etc. As the FSRD is now prepared, these counts would not be useful for prediction or assessment. (This remark

Table 2-1. Measures Based on Existing Documentation: M1 to M29

Number	Measure
M1	Pages in FSRD
M2	Paragraphs in FSRD
M3	Instances of the word "shall" in FSRD
M4	Figures in FSRD
M5	Tables in FSRD
M6	Equations in FSRD
M7	Variables in FSRD
M8	Functional Requirements
M9	Input Requirements
M10	Output Requirements
M11	Performance Requirements
M12	Subsystem Interface Requirements
M13	External Interface Requirements
M14	Operational Requirements
M15	Processing Modes
M16	Major Functions
M17	Subsystems
M18	Environmental Constraints
M19	External Interfaces
M20	TBDs
M21	Events
M22	Entities
M23	Sensor Types
M24	Sensors (total)
M25	Pages in Section 2 of FSRD
M26	Staff-Months Expended: Requirements Definition
M27	Staff-Months Expended: Requirements Analysis
M28	Number of Specification Modifications
M29	Number of Recorded Questions About Specifications

should not be construed as critical of the FSRD, which has other objectives than serving as the basis for specification measures.)

If measures M1 through M7 were calculated, the question of "includes" would need clarification. The FSRD refers to other documents that contain requirements information. The additional pages, equations, etc., should be included in the totals for the measures.

An additional concern with measures M6 and M7 is that they are specific to the type of application. Some large and complex projects may not be described by many equations or variables at the requirements stage. With equations, there is also the question of which become represented later in code. Some equations provide analytical background that is never encoded.

2.1.2 REQUIREMENTS COUNTS

Measures M8 through M14 depend on some enumeration of the requirements in the FSRD.

- M8 Number of Functional Requirements
- M9 Number of Input Requirements
- M10 Number of Output Requirements
- M11 Number of Performance Requirements
- M12 Number of Subsystem Interface Requirements
- M13 Number of External Interface Requirements
- M14 Number of Operational Requirements

With some systems, e.g., the Earth Radiation Budget Satellite Dynamics Simulator (DERBY), the requirements are numbered with a decimal system (e.g, 1.1, 1.2, ...) to express some of the detail and subsidiary concerns associated with certain requirements. The principal difficulties with all such measures are the different levels of detail and the

implicit nature of many requirements. Consider the following three numbered requirements from the DERBY FSRD (Reference 3):

- "R.1.2.3-the Profile Program will not allow users to input or modify...the area of individual spacecraft elements..." (p. 2-2)
- "R.4.3.2-the ADCS will output actuator commands to the Truth Model." (p. 5-4)
- "R.3.2.1.1-the Truth Model will interpolate the profile data using a five-point Lagrangian interpolator." (p. 4-1)

The numbering of requirements is a good practice that accommodates traceability. However, counting such requirements does not appear to be reasonable: they are expressing different levels of behavior. Also, many requirements are not written down because they are assumed to be generally known by the reader of the FSRD.

2.1.3 COUNTS OF FEATURES, CAPABILITIES, AND OMISSIONS

Measures M15 through M27 require careful interpretation of the FSRD and consistent definitions to be of any use.

<u>M15</u> Number of Processing Modes

Measure M15 can be evaluated as batch, interactive, real-time, or some other mode, depending on the application. It helps to describe the system but has a limited range of values.

<u>M16</u> Number of Major Functions

<u>M17</u> Number of Subsystems

Measure M16 needs especially consistent definition, and as potential predictors of system size, measures M16 and M17 are both probably inadequate. In a large project, each subsystem or major function may be larger than those in a small project, but the number of them may not differ. Also, with

unfamiliar applications, neither one may be known in the requirements analysis phase.

M18 Number of Environmental Constraints

To acknowledge the existence of constraints is important, but counting them is not straightforward. For example, the following statement could be counted as one, two, or three constraints:

> "The system must run in 48K on a PDP 11/70 under RSX-11."

M19 Number of External Interfaces

The awareness of interfaces is critical, but counting them requires the consistent application of rules for interpreting the contents of the FSRD. The rules would need to clarify the handling of external peripheral devices, hardware, software, and files. For example, would interfaces include only major external systems and organizational entities like the Network Control Center (NCC), Payload Operations Control Center (POCC), and Information Processing Division (IPD)? Or, would measure M19 also count the interfaces to the Graphic Executive Support System (GESS), a line printer, and a CRT display? Some of the interfaces are assumed but not explicitly identified in the FSRD (e.g., GESS).

M20 Number of TBDs

Although to-be-determined items (TBDs) are key warning signals in the early development phases, counting them can be problematical. Identifying items as TBDs implies that there is some shared understanding of how much information should be known at each point in the life cycle. For example, detailed record layouts on files may not have been determined during requirements analysis, but this lack of detail will not deter progress because the project is in an early

phase. One writer of an FSRD may identify such an item as a TBD, while another may not choose to do so because he or she

- Has a more restrictive view of a TBD item as one which is endangering progress at that point in the life cycle, or
- Has a different understanding of the information that would be known at that phase

An additional problem with identifying TBDs is that they may not be called out in the FSRD. TBDs may correspond to material omitted from the FSRD. Careful reading of the FSRD by an experienced person would be required to recognize such omissions.

M21 Number of Events
M22 Number of Entities

The rationale for measures M21 and M22 is that the behavior and environment of the system help determine its complexity. Both require consistent definitions and diligent reading of the FSRD. Entities refer to objects like sensors or momentum wheels that have data attributes. Obtaining measures M21 and M22 would be much easier with the representation described in Section 3.

M23 Number of Sensor Types
M24 Number of Sensors (total)
M25 Number of Pages in Section 2 of FSRD

Measures M23 through M25 are representative of a variety of application-specific measures that could be defined if the domain were restricted to, for example, attitude ground support systems (AGSSs). Measure M25 reflects the pattern of having Section 2 of the FSRD contain a summary of the attitude ground support requirements.

The assumption is that, if there is some consistency in the level of detail, the number of pages may serve to indicate the relative size or complexity of the AGSS.

2.1.4 RESOURCE EXPENDITURES

Measures M26 and M27 differ from the earlier measures in that they do not use the FSRD.

- <u>M26</u> Number of Staff-Months Expended: Requirements Definition
- <u>M27</u> Number of Staff-Months Expended: Requirements Analysis

They are included in this section because "existing documentation" in the title of this section is being broadly interpreted to include the SEL data base as well as hardcopy material.

Measures M26 and M27 have some attractive features. The data on staff charges to the requirements tasks are available. The rationale is that the degree of difficulty is performing the necessary analysis (as measured in staff-months) will be a predictor of the effort required to convert the product of the analysis, the FSRD, into code.

2.1.5 EVOLUTIONARY MEASURES

Measures M28 and M29, also non-FSRD data, may help to characterize the requirements.

- <u>M28</u> Number of Specification Modifications
- <u>M29</u> Number of Recorded Questions About the Requirements Specifications

Both measures would be in their early stages of collection, so it is questionable whether there would be sufficient time for the count to build up.

2.2 PRACTICALITY AND UTILITY OF THE MEASURES

Although measures M1 through M29 are based on existing documentation, it would be a mistake to conclude that no effort would be needed to extract the measures. As the comments on the measures in Section 2.1 indicate,

- Precise definitions must be formulated so the counts have some meaning
- Procedures must be established for systematically processing the FSRD to determine if the definitions are met
- Incompleteness or omissions from the FSRD must be recognized and resolved

Even if the style of the FSRD does not change, some effort will be needed in the above areas. But this effort will not guarantee that the resulting measures will be useful. The considerable latitude permitted in preparing the FSRD has the effect of reducing confidence in the measures, even when the items being counted are carefully defined.

The shortcomings of the current documentation as a basis for measurement should not be surprising. Other investigators have reported the same dissatisfaction, leading them to subjective measures as an alterative. Boehm concludes (Reference 2, p. 482):

> "Some work has been done to correlate the amount of software development effort to the number of specification elements.... These attempts have run into the same sort of definitional and normalization problems as have the 'number of routines, reports, etc'..."

Measures M26 and M27 are significantly different from the previous ones in their lack of dependence on the FSRD. Both measures seem to be promising and simple to test.

The practicality and usefulness of measures M1 through M25 may be summarized as follows:

- Extracting the measures by direct examination of the FSRD must be preceded by the development of procedures and definitions if the measures are to have any use.
- Even with careful definitions of items being counted, there is considerable doubt about the usefulness of the measures because of the latitude permitted in FSRD preparation.

Table 2-2 presents a preliminary rating of the accessibility and utility of the measures as being either high, moderate, or low. The ratings should be interpreted only as rough subjective evaluations. The comments in Section 2.1 may help explain the reasoning that led to the ratings.

Accessibility of a measure encompasses the estimated effort to

- Develop consistent definitions and procedures for extracting the data on which the measures depend
- Extract the data, with consideration for the skills and experience needed by personnel involved
- Calculate the measure

If a particular measure is computed, its estimated utility depends on

- Analyst's confidence that the measure is defined precisely enough so that it has a clear meaning
- Likelihood that the measure will relate to some important attribute of the software product or process

Table 2-2. Estimated Accessibility and Utility of Measures M1 to M29 (1 of 2)

Estimated Accessibility*	Estimated Utility*	Number	Measure
H	L	M1	Pages in FSRD
H	L	M2	Paragraphs in FSRD
H	L	M3	Instances of the word "shall" in FSRD
H	L	M4	Figures in FSRD
H	L	M5	Tables in FSRD
H	L	M6	Equations in FSRD
H	L	M7	Variables in FSRD
M	L	M8	Functional Requirements
M	L	M9	Input Requirements
M	L	M10	Output Requirements
M	L	M11	Performance Requirements
M	L	M12	Subsystem Interface Requirements
M	L	M13	External Interface Requirements
M	L	M14	Operational Requirements
M	M	M15	Processing Modes
M	M	M16	Major Functions
M	M	M17	Subsystems
M	M	M18	Environmental Constraints
M	M	M19	External Interfaces
L	M	M20	TBDs
L	M	M21	Events
L	M	M22	Entities
H	L	M23	Sensor Types

*Ratings: H = high, M = moderate, L = low.

9686

Table 2-2. Estimated Accessibility and Utility of Measures M1 to M29 (2 of 2)

Estimated Accessibility*	Estimated Utility*	Number	Measure
H	L	M24	Sensors (total)
H	L	M25	Pages in Section 2 of FSRD
H	H	M26	Staff-Months Expended: Requirements Definition
H	H	M27	Staff-Months Expended: Requirements Analysis
H	L	M28	Number of Specification Modifications
H	L	M29	Number of Recorded Questions About Specifications

*Ratings: H = high, M = moderate, L = low.

SECTION 3 - COMPOSITE SPECIFICATION MODEL

The development of the Composite Specification Model (CSM) is motivated by the limitations (noted in Section 2) in deriving useful measures from existing documentation. The remaining proposed specification measures, introduced in Section 4, are based on the CSM.

The rationale for the CSM is that no single view of a complex object should be expected to be satisfactory. The most obvious analogy is with the multiple representations used in architecture. A scale model or artist's rendering of a building, which may be appropriate to show the planning commission, is not the representation needed by the plumbers or electricians. In terms of the number of relations present, software can be more complex than buildings. A strong case has been made elsewhere that the largest software systems are the most complex objects humans have built.

If multiple perspectives are needed, are there dimensions of a system description that would enable orthogonal projections of the system onto different planes? Three such descriptive dimensions are proposed, representing the functional, contextual, and dynamic views of the system. The CSM employs these three views, choosing a particular notation to capture the perspective in each case. The expectation is that the three perspectives complement each other in providing a comprehensive understanding of a particular system. With a batch system, for example, with little input and output, the functional view may be the most meaningful, as it depicts the transformations of input quantities through intermediate stages to yield output. With the requirements for an interactive software tool, the dynamic view of the system will be the most valuable for communicating the intended operation of the system. The three views and the notation used to represent them are explained in detail in the following sections.

3.1 FUNCTIONAL VIEW

Functional processing is what the software will do--how it will transform input to produce output. The representational medium is the data flow diagram, consisting of functions that may be thought of as engines for transforming input data flows into output data flows. An attractive feature is that the reader can choose the desired level of detail in the description of functionality. Because data flow diagrams are well known, they will not be explained further. Reference 4 may be consulted for more information.

3.2 CONTEXTUAL VIEW

Unlike functional processing, which is a predictable component of most specification models, the contextual view is not so obvious a choice. This view describes the environment or information space in which the system will reside. Capturing the context of a system has been relatively undervalued as a tool for requirements engineering. A partial explanation may be that, for small programming exercises (e.g., sorting numbers or solving an equation), the background environment is either nonexistent or not a major concern; there is thus no need to try to represent it. Many of the guidelines for addressing large system development have begun as attempts to "scale-up" the approaches (e.g., structured techniques) that were successful with small programs. Because the context is not important in understanding small programs, it has not been one of the techniques that investigators pursued in this scaling-up process.

With larger systems, the context or environment is a significant element in understanding the system's behavior. The software system is modeling some portion of an environment. The system, when it is completed, will be taking its place in that environment, interacting with other objects (e.g.,

hardware, sensors, other software) that are producing behavior in the environment. To describe its behavior relative to these other objects, the system must refer to specific attributes of the objects, for example, the mean radius of the Earth or the size of fuel tanks. Likewise, events in the environment (e.g., loss of signal, thruster on-time) may trigger behavior by the system. Not all of the attributes or events in the environment are modeled by the system. In this sense, the model of the environment is not complete, nor is it ever intended to be complete. An individual attempting to understand the functioning and behavior of the software will be aided by seeing a representation of precisely those objects, attributes, and events that the system needs to know about in its environment.

The representation of the environment is not the same as a data dictionary. Data items in the dictionary may have no counterpart in the breakdown of objects, attributes, and events in the environment. Conversely, descriptors in the environment (e.g., Earth, gyro) will not always correspond to data items.

The importance of context to system understanding is receiving increasing recognition. For example, DeMarco's influential 1978 book on system specification (Reference 4) does not address the representation of context. In his 1982 book (Reference 5), however, the system's environment has been elevated to assume a major role in system specification, although the "retained-data" model he uses is not as powerful (Reference 6) as the model proposed for the CSM.

It is much more difficult to specify the requirements for large systems than to do so for small programs, because of the increase in complexity. What is the origin of the added complexity? In a rough analogy between large systems and

humans as decisionmaking, behavior-producing entities, Simon has observed (Reference 7):

> "A man, viewed as a behaving system, is quite simple. The apparent complexity of his behavior over time is largely a reflection of the complexity of the environment in which he finds himself."

The implication is that a large system is more complex because it is modeling more of a complex environment. In this sense, representing the environment in the CSM requires focusing properly on the source of the complexity.

Capturing the information space or context will be extremely valuable in making decisions about the reusability of systems. From this representation, the particular environment of an existing system will be visible. An analyst or developer will thus be able to assess the degree of reusability based on the new system's similarity to the objects, attributes, and events characterizing the environment of an existing system.

The representation of context will also help with the question of how "good" a particular requirements specification is. Two key properties that determine its "goodness" are completeness and modifiability. Making the objects and events visible through the representation of context will greatly assist any assessment of completeness. For example, if a system models thruster firings during a maneuver, "volume of fuel in tank" should be present in the representation as an attribute.

Modifiability or designing for change is a desirable attribute of a system. Its embodiment earlier in the life cycle is to "specify for change." Many of the changes to a system are responses to changes in the environment. When the specification includes a representation of the environment, the effects of such changes are easier to assess, because both

the change and the specification being changed are expressed in the same terms in the domain of the application and the user.

The form used in the CSM for representing the information space of the system is the entity-relationship (ER) approach (Reference 8). The ER model was selected because, as explained in Reference 8,

- It is among the most flexible models, having been used in a wide range of applications.
- Its view of a particular environment is more "natural" than other alternatives--requiring less effort to apply.
- It is a sufficiently comprehensive to serve as a framework for deriving the three more-specific models: network, relational, and entity-set.

Four terms will be used in the ER view: entities, relationships, attributes, and value sets. Brief definitions and examples will be presented for each term. Reference 8 contains a more thorough introduction.

Entities are identifiable objects in the environment. Some examples are a momentum wheel, a user, a CRT display, a fuel tank, Earth, and the spacecraft. Events (e.g., start of maneuver, end of integration step) are considered to be entities in the ER approach. In the CSM, the entities that correspond to events can be identified separately but share all of the properties of entities. In the following discussion, entities may includes events.

Relationships are associations among entities and are defined as are relations in discrete mathematics (Reference 9). Examples of relations are Earth-spacecraft and fuel tank-initial loading time.

Information about entities and relationships is expressed by a set of attribute-value pairs. An attribute is a property or feature of the entity or relationship. For example, the entity "fuel tank" may have attributes of volume and location on the spacecraft.

Value sets combine the concepts of the units of measure with ranges and types of acceptable values for attributes. For example, the value set for the attribute "volume" may be "INCHES3: cubic inches, nonnegative real range."

A valuable conceptual feature of the ER approach is the ability to associate attributes with relationships as well as entities. As an example, the attribute "unit vector from the spacecraft center of mass to the center of Earth" is associated with the "Earth-spacecraft" relationship. It would be inaccurate to associate with the entities "Earth" or "spacecraft."

3.3 DYNAMIC VIEW

The third component of the CSM is the dynamic view, representing the behavior of the system over time. The notation used is the state transition diagram, a directed graph in which the nodes correspond to states of the system and the directed arcs show the possible changes in state (Reference 5). Events in the environment (e.g., a user selects a menu option) provide the stimuli to trigger a state change.

SECTION 4 - MEASURES BASED ON THE COMPOSITE SPECIFICATION MODEL

This section defines measures based on the Composite Specification Model introduced in Section 3. The measures are grouped according to the three dimensions of the CSM: functional view, contextual view, and dynamic view. All of the measures defined in this section are listed in Table 4-1.

4.1 FUNCTIONAL VIEW: DATA FLOW DIAGRAMS

Data flow diagrams are presented at different levels of detail. The measures should be based on the most detailed, fully decomposed data flow diagram so that there will be some consistency and opportunity for comparison between projects.

Although objective counts will be available from the completed data flow diagram, the process of developing the data flow diagram has a subjective component. Personal judgment is involved in deciding how to decompose system requirements into function nodes and data flows. The structured analysis methodology and its derivatives recognize four methods for deciding when the lowest level partitioning has been reached (References 4 and 5):

- When the minispecification is approximately one page in length
- When the function has a single input data flow and a single output data flow
- When application of Jackson's concept of boundary clashes indicates that the partitioning should stop
- When further partitioning will not reduce the average number of data items on input or output data flows around function nodes

Table 4-1. Specification Measures Based on CSM (1 of 2)

Number	Measure
M101*	Functional Primitives
M102*	Longest Path
M103*	Interface Count
M104*	System In-Arcs
M105*	System Out-Arcs
M106*	File In-Arcs
M107*	File Out-Arcs
M108*	Internal Arcs
M109	Maximum Internal Arcs
M110	Arc Density
M111	Relative Arc Density
M112	System In/Out-Arcs
M113	File In/Out-Arcs
M114	Total In-Arcs
M115	Total Out-Arcs
M116	Total System Non-File Arcs
M117	Total Internal and File Arcs
M118	Total Arcs
M119*	System In-Data Items
M120*	System Out-Data Items
M121*	File In-Data Items
M122*	File Out-Data Items
M123*	Internal Data Items
M124	System In/Out-Data Items
M125	File In/Out-Data Items
M126	Total In-Data Items
M127	Total Out-Data Items
M128	Total System Non-File Data Items
M129	Total Internal and File Data Items

*Denotes basic measure, i.e., not computable from other measures on this list.

Table 4-1. Specification Measures Based on CSM (2 of 2)

Number	Measure
M130	Total Data Items
M131	Internal Arc Weight
M132	System In/Out-Arc Weight
M133	File In/Out-Arc Weight
M134	System Non-File Arc Weight
M135	Internal and File Arc Weight
M136	Arc Weight
M137	Classification Measure #1
M138	Classification Measure #2
M139	Classification Measure #3
M140	Classification Measure #4
M141	Classification Measure #5
M142*	Derivation Set Complexity
M143*	Relative Derivation Set Complexity
M144*	Weighted Function
M201*	Entities
M202*	Events
M203*	Relations
M204*	Attributes
M205*	Value Sets
M206	Relation Density
M207	Attribute Density
M208	Value Set Density
M301*	States
M302*	Transitions
M303	Activity
M401	Functional/Contextual Ratio
M402	Functional/Dynamic Ratio
M403	Dynamic/Contextual Ratio

*Denotes basic measure, i.e., not computable from other measures on this list.

The application of the CSM to a real system in Section 5 confirmed that following one guideline may violate another one: human judgment remains a factor in the decomposition decision.

Several of the measures defined in this section are minor variations of one another. It is not recommended that all of the measures be used. They are included to illustrate the details that must be considered when such measures are defined.

A further distinction may be made between raw counts and derived measures, which combine the raw counts in various ways. These derived measures are included because they are analogs of size and complexity measures that have been proposed by other investigators. For example, several early complexity measures (e.g., Reference 10) have been based on some count of input and output. Some proposed measures in this section are therefore similarly defined.

Figure 4-1 is an example of a data flow diagram consisting of six nodes and four external entities (denoted I, II, III, and IV). Table 4-2 gives the values of the basic measures extracted from the example.

4.1.1 FUNCTIONS AND ENTITIES: BASIC COUNTS (MEASURES M101 TO M103)

<u>M101</u> Functional Primitives

This is the number of nodes in a fully decomposed data flow diagram.

<u>M102</u> Longest Path

This is the number of nodes in the longest path from any source external entity to any sink external entity. It is valid only if loop-free data flow diagrams are enforced.

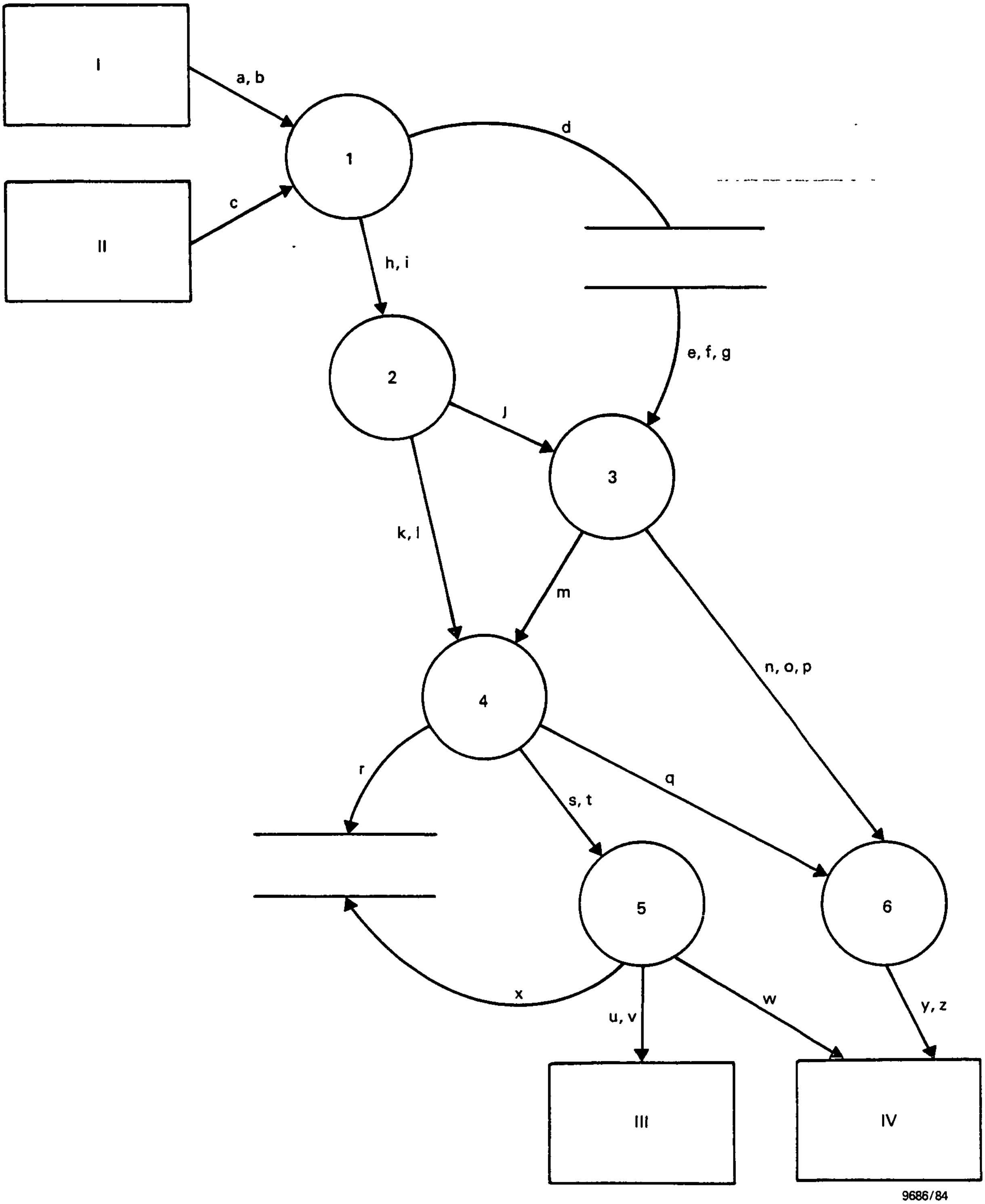


Figure 4-1. Example of Data Flow Diagram

Table 4-2. Basic Measures Extracted From Example in Figure 4-1

Measure Number	Measure Name	Value
M101	Functional Primitives	6
M102	Longest Path	5
M103	Interface Count	4
M104	System In-Arcs	2
M105	System Out-Arcs	3
M106	File In-Arcs	2
M107	File Out-Arcs	2
M108	Internal Arcs	7
M119	System In-Data Items	3
M120	System Out-Data Items	5
M121	File In-Data Items	4
M122	File Out-Data Items	2
M123	Internal Data Items	12

__M103__ Interface Count

This is the number of external entities that have data flows to/from the system. It is not necessarily equal to the count of arcs to/from external entities.

4.1.2 ARCS: BASIC COUNTS (MEASURES M104 TO M108)

__M104__ System In-Arcs

This is the number of arcs from outside entities to the system. It does not include arcs from files.

__M105__ System Out-Arcs

This is the number of arcs from the system to outside entities. It does not include arcs to files.

__M106__ File In-Arcs

This is the number of arcs from files.

__M107__ File Out-Arcs

This is the number of arcs to files.

__M108__ Internal Arcs

This is the number of arcs between nodes that are part of the system. It does not include arcs to/from files or to/from external entities.

4.1.3 ARCS: DERIVED MEASURES (MEASURES M109 TO M118)

__M109__ Maximum Internal Arcs

This measure is obtained from measure M101, number of functional primitives. It assumes that a fully decomposed data flow diagram is loop-free. It is derived as follows:

$$M109 = ((M101)^2 - M101)/2$$

M110 Arc Density

This measure is obtained from measure M108, number of internal arcs, and measure M101, number of functional primitives. It is derived as follows:

M110 = M108/M101

M111 Relative Arc Density

This measure is obtained from measure M108, number of internal arcs, and measure M109, maximum number of internal arcs. It is derived as follows:

M111 = M108/M109

M112 System In/Out-Arcs

This is the number of arcs in/out of the system. It is derived as follows:

M112 = M104 + M105

M113 File In/Out-Arcs

This is the number of arcs in/out of files. It is derived as follows:

M113 = M106 + M107

M114 Total In-Arcs

This is the number of arcs input from external entities or from files. It is derived as follows:

M114 = M104 + M106

M115 Total Out-Arcs

This is the number of arcs output to external entities or to files. It is derived as follows:

M115 = M105 + M107

M116 Total System Non-File Arcs

This is the number of internal and system input/output arcs, excluding to/from files. It is derived as follows:

$$M116 = M108 + M112$$

M117 Total Internal and File Arcs

This is the number of internal arcs and arcs to/from files. It is derived as follows:

$$M117 = M108 + M113$$

M118 Total Arcs

This is the total number of arcs: internal, to/from files, and to/from external entities. It is derived as follows:

$$M118 = M108 + M112 + M113$$

4.1.4 DATA ITEMS: BASIC COUNTS (MEASURES M119 TO M123)

M119 System In-Data Items

This is the number of distinct data items input to the system from external entities. It does not include data items from files.

M120 System Out-Data Items

This is the number of distinct data items output to external entities from the system. It does not include data items to files.

M121 File In-Data Items

This is the number of distinct data items input from files.

M122 File Out-Data Items

This is the number of distinct data items output to files.

M123 Internal Data Items

This is the number of distinct data items on internal arcs. It does not include data items on arcs to/from files.

4.1.5 DATA ITEMS: DERIVED MEASURES (MEASURES M124 TO M130)

<u>M124</u> System In/Out-Data Items

This is the number of distinct data items input or output. It is derived as follows:

$$M124 = M119 + M120$$

<u>M125</u> File In/Out-Data Items

This is the number of distinct data items to/from files. It is derived as follows:

$$M125 = M121 + M122$$

<u>M126</u> Total In-Data Items

This is the number of distinct data items input from external entities or files. It is derived as follows:

$$M126 = M119 + M121$$

<u>M127</u> Total Out-Data Items

This is the number of distinct data items output to external entities or files. It is derived as follows:

$$M127 = M120 + M122$$

<u>M128</u> Total System Non-File Data Items

This is the number of distinct internal and system input/output data items excluding to/from files. It is derived as follows:

$$M128 = M123 + M124$$

<u>M129</u> Total Internal and File Data Items

This is the number of distinct internal and file data items. It is derived as follows:

$$M129 = M123 + M125$$

M130 Total Data Items

This is the number of distinct data items appearing on all arcs: internal, to/from files, and to/from external entities. It is derived as follows:

M130 = M123 + M124 + M125

4.1.6 ARC WEIGHTS (MEASURES M131 TO M136)

M131 Internal Arc Weight

This measure is obtained from measure M123, internal data items, and measure M108, internal arcs. It is derived as follows:

M131 = M123/M108

M132 System In/Out-Arc Weight

The measure is obtained from measure M124, system in/out-data items, and measure M112, system in/out-arcs. It is derived as follows:

M131 = M124/M112

M133 File In/Out-Arc Weight

This measure is obtained from measure M125, file in/out-data items, and measure M113, file in/out-arcs. It is derived as follows:

M133 = M125/M113

M134 System Non-File Arc Weight

This measure is obtained from measure M128, total system non-file data items, and measure M116, total system non-file arcs. It is derived as follows:

M134 = M128/M116

M135 Internal and File Arc Weight

This measure is obtained from measure M129, total internal and file data items, and measure M117, total internal and file arcs. It is derived as follows:

M135 = M129/M117

M136 Arc Weight

This measure is obtained from measure M130, total data items, and measure M118, total arcs. It is derived as follows:

M136 = M130/M118

4.1.7 MEASURES FOR EARLY CLASSIFICATION (MEASURES M137 TO M141)

Measures M137 through M141 are analogous to proposals for classification of projects. Low values imply "function-strong" applications; high values imply "data-strong" applications.

M137 Classification Measure 1

This measure is obtained from measure M120, system out-data items, and measure M101, functional primitives. It is derived as follows:

M137 = M120/M101

M138 Classification Measure 2

This measure is obtained from measure M127, total out-data items, and measure M101, functional primitives. It is derived as follows:

M138 = M127/M101

M139 Classification Measure 3

This measure is obtained from measure M124, system in/out-data items, and measure M101, functional primitives. It is derived as follows:

M139 = M124/M101

M140 Classification Measure 4

This measure is obtained from measure M128, total system non-file data items, and measure M101, functional primitives. It is derived as follows:

$$M140 = M128/M101$$

M141 Classification Measure 5

This measure is obtained from measure M130, total data items, and measure M101, functional primitives. It is derived as follows:

$$M141 = M130/M101$$

4.1.8 ANALYTIC MEASURES (MEASURES M142 AND M143)

M142 Derivation Set Complexity

M143 Relative Derivation Set Complexity

Measures M142 and M143 could provide predictions of effort or complexity. They have been adapted from Reference 11, which uses control information as well as data flow. The results could be modified so control flow is not needed.

The principle behind these measures is to focus on the size and complexity of the set of data items that derive each output data item. The measures are time-consuming to compute manually, but the process could be automated.

4.1.9 AGGREGATE MEASURES (MEASURE M144)

Another approach to project-level measures is to build them from counts associated with each functional primitive node in the data flow diagram: number of arcs in and out, number of data items in and out, etc. Instead of simply adding the counts for each node, the values could also be weighted. If the Section 2 measure "number of equations" was developed

and associated with each function node, it could be used as a weighting for a measure of functionality of each node, such as

(number of equations) x (number of data items in and out)

Summing the node-level measures will yield project-level measures, which will generally differ from the measures presented earlier in this section. For example, measure M123, number of internal data items, will not generally have the same value as an aggregate measure formed by adding the data items surrounding each node, because of duplicate counting in the latter case. The use of aggregate measures would resemble the approach used by Albrecht (Reference 12) for commercial applications.

One aggregate measure, M144, weighted function, has been proposed by DeMarco (Reference 5) as an early estimate of system size:

Define DE_i as the number of data items "around" node i, i.e., the number of data items appearing on any arcs into or out of node i.

For each node i, compute

$$(DE_i \times \log_2 (DE_i))/2$$

<u>M144</u> Weighted Function

This measure is derived as follows:

$$M144 = \sum_{i=1}^{M101} ((DE_i \times \log_2 (DE_i))/2$$

where M101 is the number of nodes (functional primitives).

The entire set of aggregate measures will not be presented in this section. They would be defined by modifying the

definitions used in measures M101 through M141 such that the counts are made on each node and then added together.

4.2 CONTEXTUAL VIEW: ENTITY-RELATIONSHIP REPRESENTATION (MEASURES M201 TO M208)

Measures of the information space of the system are based on counts of the principal constituents of the entity-relationship description: entities, events, relations, attributes, and value sets. The only variation from the definitions of the terms in Section 3 is to maintain a separate count of events and a count of other (nonevent) entities.

M201 Entities

This is a count of the distinct nonevent entities about which the system must have some knowledge.

M202 Events

This is a count of the distinct environmental occurrences that affect the behavior of the system.

M203 Relations

This is a count of the distinct relations between collections of entities or events. Relations are counted when attributes are associated with them.

M204 Attributes

This is a count of the distinct properties or characteristics that must be known to the system about the entities, events, or relations.

M205 Value Sets

This is a count of the distinct collections of attribute values, incorporating concepts of range, type, and units of measure.

M206 Relation Density

The maximum number of relations is 2^E, where E = M201 + M202. Measure M206, how many of the possible relations exist, is derived as follows:

$$M206 = M203/(2^E)$$

This measure will range between 0 and 1 and may assist in characterizing the degree of interconnectedness of the information space.

M207 Attribute Density

This is the mean number of attributes per entity or event. It is derived as follows:

$$M207 = M204/E$$

where E = M201 + M202.

Measure M207 may assist in characterizing the depth of knowledge that the system must maintain.

M208 Value Set Density

This is defined as the number of value sets relative to the number of attributes. It is derived as follows:

$$M208 = M205/M204$$

Measures M208 will range between 0 and 1. A low measure (near zero) indicates that relatively few distinct sets of values are being used to represent attribute values; a high measure (near unity) indicates that nearly every attribute has its own distinct value set.

This measure may assist in measuring a suspected highly error-prone activity, that of accommodating all of the various coordinate systems, frames of reference, units, and acceptable ranges.

4.3 DYNAMIC VIEW: STATE TRANSITION DIAGRAMS (MEASURES M301 TO M303)

M301 States

M302 Transitions

Measures of the dynamic view are based on counts of these two elements that make up the diagrams.

M303 Activity

Measure M303 relates measures M301 and M302 to capture the level of activity in the dynamic view.

It is a measure of the actual number of transitions relative to the maximum number possible. It is derived as follows:

$$M303 = M302/(M301)^2$$

Measure M303 will range between 0 and 1. It allows transitions from a state to itself.

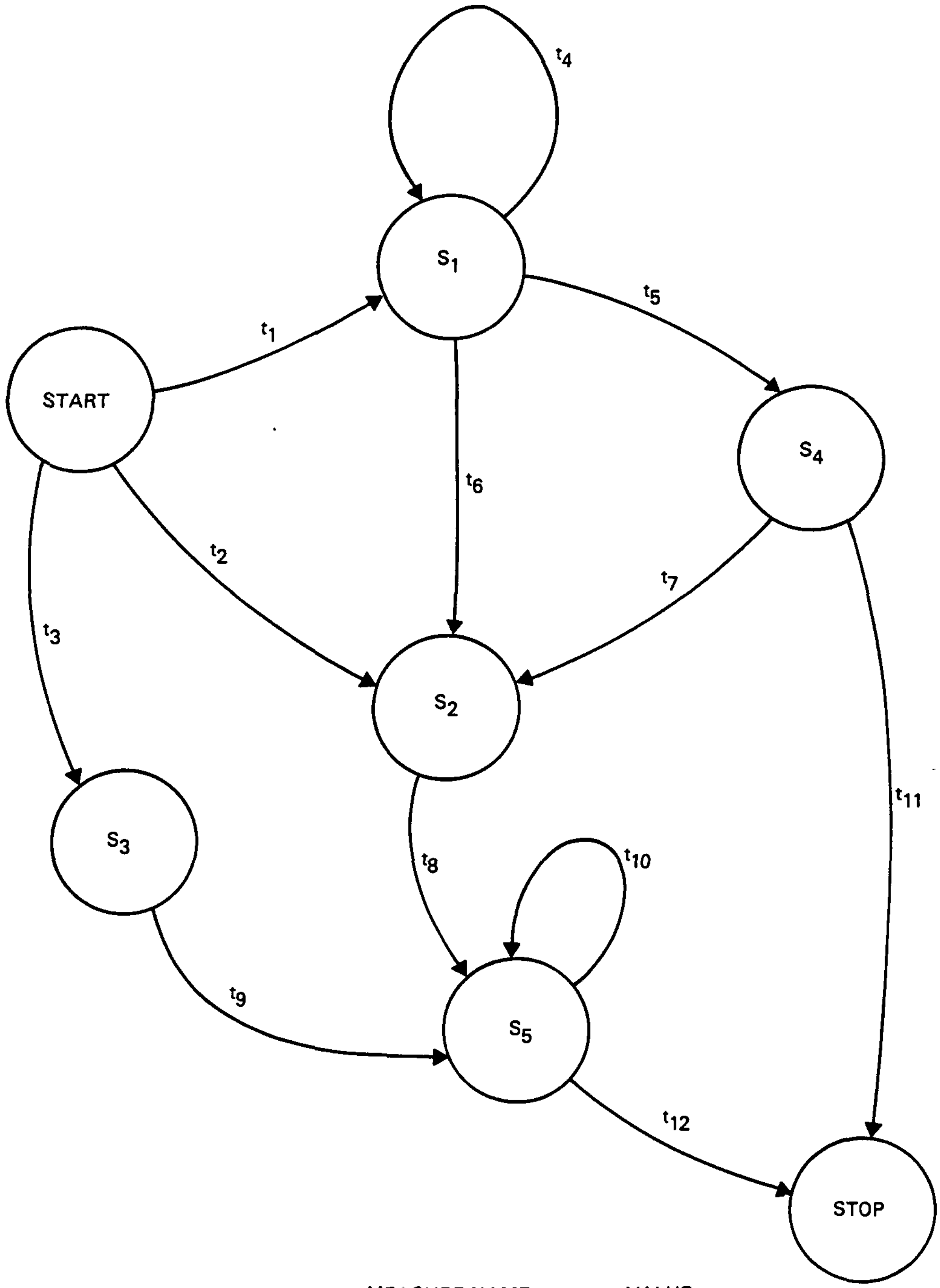
Figure 4-2 is an example of a state transition diagram with values extracted for M301 and M302.

4.4 MEASURING THE DOMINANT VIEW (MEASURES M401 TO M403)

The counts from separate views can be combined in various ways to yield a fourth group of measures from the CSM. Measures M401 through M403 could be used to assess the relative strength of each of the three views. Experience with the measures may allow the early characterization of a system, for example, "function-strong" or "context-strong," because of the dominant role of a particular view.

One approach is to select, from each view, a measure that represents that view. The ratios of these three measures would capture the notion of relative dominance, for example,

- Functional view--Measure M101, functional primitives
- Contextual view--Measure M203, relations
- Dynamic view--Measure M301, states



	MEASURE NAME	VALUE
M301	NUMBER OF STATES	7
M302	NUMBER OF TRANSITIONS	12

9686/84

Figure 4-2. Example of State Transition Diagram

M401 Functional/Contextual Ratio

This measure is derived as follows:

M401 = M101/M203

M402 Functional/Dynamic Ratio

This measure is derived as follows:

M402 = M101/M301

M403 Dynamic/Contextual Ratio

This measure is derived as follows:

M403 = M301/M203

If such measures are used on several projects, patterns may emerge, allowing the identification of threshold values and the classification of related systems.

SECTION 5 - AN EXERCISE IN EXTRACTING SPECIFICATION MEASURES

This section reports on an exercise in extracting specification measures from a real system. Section 5.1 identifies 16 specification measures recommended for use in the SEL. Section 5.2 presents the results of extracting these 16 measures from flight dynamics software requirements.

5.1 RECOMMENDED SPECIFICATION MEASURES FOR EXTRACTION

Table 5-1 lists a recommended set of 16 specification measures. The first two measures are available from existing documentation and are defined in Section 2. The remaining measures are defined in Section 4 and are derived from the CSM representation of requirements. Except for the analytic measure, weighted function, all the CSM-based measures are explicit counts. As discussed in Section 4, these explicit counts enable the calculation of more derived measures from this basic set.

The following criteria were used to select the 16 measures in Table 5-1 from the 87 measures defined in Sections 2 and 4:

- Availability and accessibility of the measures, including ease of extraction
- Usefulness of the measures as indicated by their potential to measure some key properties of the requirements
- Ability of the measures to be combined with other measures (e.g., as normalization factors), thereby creating potentially useful analytic measures

5.2 RESULTS OF EXTRACTING SPECIFICATION MEASURES

As an exercise, the measures from Table 5-1 were extracted from the requirements for a real system. This section

Table 5-1. Recommended Set of Specification Measures

Measure Number[a]	Measure Name
M26	Staff-Months Expended: Requirements Definition
M27	Staff-Months Expended: Requirements Analysis
M101	Functional Primitives
M103	Interface Count
M108	Internal Arcs
M123	Internal Data Items
M124	System In/Out Data Items
M125	File In/Out Data Items
M144	Weighted Function
M201	Entities
M202	Events
M203	Relations
M204	Attributes
M205	Value Sets
M301	States
M302	Transitions

[a]As defined in Sections 2 and 4.

summarizes the measurement exercise, which is described in more detail in Reference 13.

Specification measures were extracted from the Yaw Maneuver Control Utility (YMCU) of the Earth Radiation Budget Satellite (ERBS). The YMCU software consists of 11,191 source lines of FORTRAN.

The results of the measurement exercise are summarized in two parts:

- The actual values extracted
- The measurement process

The actual values extracted from the requirements and listed in Table 5-2 constitute only a single data point in any attempt to draw inferences from the values. Specification measures must be extracted for several projects before any patterns or trends might possibly emerge.

The exercise demonstrated that the measurement process is feasible. The process involved the preparation of the CSM for the YMCU software. This alternative representation of requirements (Reference 14) not only facilitates measurement but also serves as a clearer reference document.

The extraction process required 1.7 staff-months of effort, almost all of which was spent understanding the requirements and recasting them according to the CSM. Only 9 hours were needed to calculate the measures from the CSM representation.

Table 5-2. Computed Metric Values for ERBS YMCU

Metric Number[a]	Metric Name	Value
M26	Staff-Months Expended: Requirements Definition	NA[b]
M27	Staff-Months Expended: Requirements Analysis	2.1
M101	Functional Primitives	39
M103	Interface Count	3
M108	Internal Arcs	60
M123	Internal Data Items	42
M124	System In/Out Data Items	67
M125	File In/Out Data Items	74
M144	Weighted Function	688
M201	Entities	11
M202	Events	14
M203	Relations	19
M204	Attributes	91
M205	Value Sets	29
M301	States	7
M302	Transitions	11

[a]From Sections 2 and 4.

[b]Not available; see Reference 13 for discussion.

SECTION 6 - ASSESSMENT OF THE SPECIFICATION MEASURES STUDY

This section presents an assessment of the investigation into specification measures for use by the SEL. This assessment covers the strengths and weaknesses of the defined specification measures, the CSM, and the implications of the metrics extraction exercise. The results of the investigation are outlined as they relate to the study objective, and directions for future studies and applications of specification measures are proposed.

6.1 STRENGTHS AND WEAKNESSES OF THE SPECIFICATION MEASURES

The strengths of the set of specification measures lie in four areas:

- Quantity
- Objectivity
- Breadth
- Extractability

The quantity of measures defined is viewed as a strength. Requirements definition and analysis have been difficult phases in which to introduce measurement. With the definition of 87 measures, there are increased opportunities for the emergence of useful indicators of key properties of the requirements.

The measures are objective, either explicit counts or well-defined calculations based on the counts. A particular requirements representation will yield a single set of measurements. Assuming that no errors are made in the extraction procedure, the measurements will be unaffected by the analyst extracting the measures.

The specification measures exhibit greater breadth than those reported in the literature. Historically, specification efforts have been directed to describing the required

functionality of the system. Measures of the specification have reflected this orientation. In the current study, measures are defined for the system's environment and its dynamic behavior, as well as its functionality.

The measures are easy to extract from the requirements representation. Most of the measures are explicit counts, and the remainder are analytic measures that can be computed by hand calculation.

The weaknesses in the set of specification measures lie in three areas:

- Lingering effect of human judgment
- Inability to measure some properties of requirements
- Limited use as "stand-alone" measures

Human judgment is still a factor in the recommended specification measures. Although the measures are objective, they are extracted from a representation that relies on subjective judgment. For example, as noted in Section 4, several guidelines have been proposed to help analysts decide when to stop decomposing processes during data flow analysis (Reference 5). Depending on which guideline is followed, the data flow diagrams may be expected to differ even when the same requirements are being analyzed. The measures based on the data flow diagrams will, of course, differ as well.

It should not be surprising that the role of human judgment has not been eliminated by these specification measures. Requirements analysis, being the first life cycle phase, occurs at a time when there is the most uncertainty about the needs of the system. It seems reasonable that any measures during this phase would reflect this uncertainty.

Specification measures are not the only measures influenced by subjective considerations. Human judgment continues to affect measures that are widely considered to be among the most objective available anytime during the life cycle. Lines of code, for example, is a measure that is influenced by the coding style of the individual programmer.

A second perceived weakness is the inability of the measures to address certain properties of requirements. It is not clear that, even after applying these measures on several projects, measures will emerge for consistency, completeness, or understandability. Rough, relative, subjective measures of such properties may result from the use of questionnaires. For example, several peer analysts would be asked to rate each property on a scale from one to five. When a group of projects is scored in this way, rating patterns may be used as a reference for assessing the properties on a new project (Reference 15). The current study has not pursued questionnaires because of the strong subjective nature of any resulting measures.

Although measures for consistency, completeness, and understandability are not forthcoming, the CSM representation has contributed in this regard. It is found (Section 6.2) to be a better medium than the requirements document for identifying inconsistency and incompleteness.

A third weakness of the set of specification measures is the limited ability for a single project to be useful without reference to other projects. This weakness was demonstrated in the extraction exercise of Section 5. The magnitude of the numbers did not convey any message about the requirements being large or small, easy or difficult, etc. The measures will have meaning only when other projects are measured and their corresponding concluding values (size, error rate, etc.) are recorded.

To a degree, this weakness is unavoidable. Measures associated with implementation and testing are more meaningful than those associated with specification and design. The activities of coding and debugging are familiar to programmers. The system product, expressed as source lines of code, has a degree of visibility that is not found in the early life cycle phases. This weakness may be expected to persist with specification measures generally until the products and process of requirements analysis become regularized.

6.2 STRENGTHS AND WEAKNESSES OF THE CSM

The CSM has both strengths and weaknesses. Its perceived strengths lie in the following areas:

- Facilitating objective specification measures
- Supporting multiple views of requirements
- Presenting requirements in a graphical, nonnarrative style
- Providing early capture of key requirements information
- Facilitating reusability and maintainability
- Evaluating properties of requirements
- Anticipating object-oriented design

The CSM facilitates the definition and extraction of objective specification measures. It was developed in response to the need for such measures, and Section 4 demonstrates that 58 measures are indeed definable due to the CSM.

The multiple viewpoints of the CSM provide breadth in the coverage of requirements. The CSM accommodates the different needs of users who may have a particular interest,

for example, in the dynamic behavior of the system. The CSM shifts the attention of requirements analysts from purely functional concerns to consideration of context and state transition. The discipline of developing each viewpoint provokes insights that aid the analyst in understanding the requirements.

Through its use of a largely nonnarrative style, the CSM is a more accessible form of requirements expression. The graphs and lists contribute to the understandability of the requirements. The CSM structure exposes the objects, relations, states, and functions. This visibility enhances traceability, maintainability, and early configuration control.

By capturing more of the requirements earlier, the CSM reduces the incidence of costly "rediscovery" of key information. Value sets are examples of important data that were not recognizable as distinct objects in non-CSM representations. Value sets have the flexibility to capture all of the important characteristics of the values that attributes may assume. As noted in Section 3, they encompass the concepts of type, structure, and unit of measure. Value sets can, however, be extended to include additional characteristics: coordinate system, precision, and range. Uncertainty about the correct value set leads to coding errors. By capturing the value sets early, the CSM serves as a valuable requirements data base.

The CSM supports the organizational objective of reusability of software products. The benefits of reusability are magnified when it is applied at earlier phases in the life cycle. By its contextual view, the CSM captures the problem domain of the software. Analysts can see the objects and relations that are being modeled so that opportunities for

reuse are easier to detect. Because many enhancements to a system are due to changes in the environment, the maintainer can work directly with the contextual model in the CSM to record those changes.

The CSM facilitates the evaluation of requirements properties. Requirements specifications serve a diverse audience--analysts, developers, managers, and customers--who want to determine the degree to which the requirements possess certain properties. The CSM makes it easier to identify inconsistency, for example, because of its multiple complementary views and its graphical style. Likewise, software complexity may be assessed by considering the state behavior along with the interconnection of entities shown by the relations.

Through its identification of entities and attributes, the CSM serves as a logical predecessor of object-oriented design. Developers who use the CSM will find it to be a good starting point as they encapsulate objects and their operations into logical units for design.

The weaknesses of the CSM lie in three areas:

- Labor intensity
- Lack of integration of views
- Incompleteness

The CSM is highly labor intensive, relying on the efforts of analysts to understand the requirements and cast them in CSM form. Portions of the CSM can be assisted by automation. The extraction exercise used the Index Technology Excelerator workstation (Reference 16) to support the functional view, i.e., drawing data flow diagrams and maintaining a data dictionary. Graphics and word processing software can help the analyst with the two other views as well. The analyst using the CSM does not, however, obtain the reports and consistency checking that would be produced automatically from the use of a specification language system. Such systems were

explicitly not pursued because of the recommendations from previous SEL studies (Reference 17).

The multiple views of the CSM are distinct, requiring additional effort to integrate them. With the YMCU software, an extra table was prepared to show the relationship between the functional and dynamic views, specifically to identify the functional processes that are active during every software state (Reference 14).

The completeness of the CSM is an issue if it is contemplated as an alternative to the functional specification and requirements document. Two weaknesses of the CSM regarding its completeness are its provision for storing mathematical equations and its scope.

Mathematical equations are used to specify flight dynamics software requirements. They can be accommodated in the CSM by using a process description that can be associated with every functional primitive in the data flow analysis. The descriptions will include the equations required to specify how the input data flows will be transformed into output data flows.

The weakness of the CSM's scope is due to the broad definition of requirement that has been used in this investigation:

> "A requirement is any property of the proposed system that determines its acceptability."

This definition may be contrasted with a narrower definition of a requirement as "what the software does--its functionality." With such restricted definitions, the CSM has no weakness in its scope. With the broad, more realistic definition, there are many examples of requirements that could not be represented in the CSM:

- The system must respond in less than 5 seconds.
- Both batch and terminal use must be supported.
- The output can be tape or disk.

With natural language descriptions of requirements, there is no need to recast such examples in a restricted notation or style. The CSM may be extended to include more views (e.g., an operational view to represent some performance requirements) but employing such a broad definition of requirement guarantees that some requirements will remain outside the scope of the CSM. A list of such requirements will be a necessary accompaniment to the CSM for it to serve as a complete specification.

6.3 IMPLICATIONS OF THE METRICS EXTRACTION EXERCISE

The results of the metrics extraction exercise (presented in Reference 13) are used here to consider the effort (greater than or less than that required in the exercise) that may be required to measure future projects. Reference 13 reported 1.7 staff-months spent by an analyst to recast the requirements into the CSM and extract the measures. For comparison, 2.1 staff-months were spent by the developer to perform the requirements analysis.

Four factors that influenced the effort expenditures in the exercise will continue to affect the effort required on future projects:

- Staff experience with the application
- Staff experience with the CSM
- Effort reporting practices
- Availability of requirements information

The first factor affecting effort is the staff experience with the application. The exercise involved a relatively small system, the Yaw Maneuver Control Utility, consisting of 11,191 source lines of FORTRAN. The requirements analysis and the CSM representation were performed by single (different) individuals. As a result, the effort values were more sensitive to individual differences than would be true of larger projects with a team approach.

The analyst preparing the CSM representation had no prior experience with flight dynamics applications. This inexperience undoubtedly lengthened the time necessary for the analyst to understand the requirements. The analyst was, however, very experienced with the components of the CSM. This familiarity obviated the need for CSM training. Future use of the CSM will probably require such training.

The reported effort of 1.7 staff-months is sensitive to the practices employed for collecting effort data. During the exercise, time was charged only when the analyst was working specifically on the exercise. For example, time spent defining the measures or reading the literature on specification measures was not charged to the measurement exercise.

The availability of requirements information affected the effort expended on the exercise and will affect the effort required on future uses of the CSM. Because the exercise was a retrospective study, the YMCU source code was available. As Reference 13 explains, the code was used selectively to resolve questions about the requirements. Referring to the code was faster than asking the developer for assistance. However, the exercise was slowed because the code provided requirements information at a very detailed level. The analyst had to filter out the excessive detail, consistent with the intention of the exercise to depend only on information potentially available at requirements time.

6.4 <u>RECONSIDERATION OF THE STUDY OBJECTIVE</u>

The objective of the study (Section 1.1) was to investigate measures that provide a quantitative characterization of the size and nature of the software requirements. The study has succeeded in defining a wide range of objective measures and recommending a comprehensive subset for use.

Progress was also made in addressing three questions posed in Section 1.1 regarding the behavior, size, and cost of a proposed system. Through the CSM representation, greater likelihood exists for capturing essential aspects of system behavior. The estimation of the size and cost of a system is enhanced by using the measures recommended in this study for functional primitives, weighted function, relation density, states, and transitions. The discussions in Sections 6.1 through 6.3 offered additional assessments of the degree to which study objectives have been met.

6.5 FUTURE DIRECTIONS FOR SPECIFICATION MEASURES IN THE SEL

An obvious alternative for future development is to apply the metric extraction procedure to more projects. As explained in Section 5, only when more projects are measured can patterns emerge from the extracted metric values. Future use of the specification measures will also help refine the CSM and the extraction procedure. Additional benefits would accrue if the target for measurement were a new system, not one (like the YMCU) that is already developed. It would eliminate the difficulty experienced during the exercise in trying to identify which information would have been available at an earlier time.

A more immediate plan is to solicit comments on the recast requirements for the YMCU (Reference 14). Such responses from requirements users (analysts, developers, managers, and customers) will help determine the potential usefulness of the CSM representation.

Continued monitoring of research by outside organizations on specification measures is a possible future activity. Potentially promising developments can be interpreted for adaptation by the SEL.

These suggested future directions share a desire to build on the results of this study for the improvement of the software development process in the flight dynamics environment.

APPENDIX - ERRORPRONE CHARACTERISTICS OF REQUIREMENTS

As part of the analysis to identify what should be measured, a brief study of requirements errors was conducted. The intent was to uncover particular aspects of the requirements that are leading to errors, so that measures can be defined to address those aspects. The Software Engineering Laboratory (SEL) file of Change Report Forms (CRFs) was scanned for changes due to requirements errors or specifications errors on all projects as of March 22, 1984. The search identified 34 forms reporting requirements errors and 126 reporting specifications errors. The affected forms were examined to try to detect if a few types of mistakes explained a high percentage of the errors. No such relationship was found. In fact, from the comments on the form, it was difficult in most cases to understand why the error was identified as being one in requirements or specifications. A possible explanation was suggested by an SEL colleague: At the level of the individual programmer who completes the CRF, the programming assignment received from the task leader may be interpreted as constituting the requirement or specification. When there is a change in that assignment, it is reported as a requirement or specification error, regardless of its relationship to requirements documentation.

REFERENCES

1. Software Engineering Laboratory, SEL-84-001, Manager's Handbook for Software Development, W. W. Agresti, F. E. McGarry, D. N. Card et al., April 1984

2. B. W. Boehm, Software Engineering Economics, Englewood Cliffs, N.J.: Prentice-Hall, 1981

3. Computer Sciences Corporation, CSC/TM-83/6052, Earth Radiation Budget Satellite (ERBS) Dynamics Simulator Requirements and Mathematical Specifications, July 1983

4. T. DeMarco, Structured Analysis and System Specification, New York: Yourdon, Inc., 1978

5. T. DeMarco, Controlling Software Projects, New York: Yourdon Press, 1982

6. W. W. Agresti, "Review: Controlling Software Projects by T. DeMarco," ACM Computing Reviews, September 1983

7. H. Simon, The Sciences of the Artificial, Cambridge, Mass.: M.I.T. Press, 1970

8. P. Chen, "The Entity-Relationship Model--Toward a Unified View of Data," ACM Transactions on Data Base Systems, March 1976

9. C. L. Liu, Elements of Discrete Mathematics, New York: McGraw-Hill, 1977

10. S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," IEEE Transactions on Software Engineering, September 1981

11. M. H. Whitworth and P. A. Szulewski, "The Measurement of Control and Data Flow Complexity in Software Designs," Proceedings, IEEE COMPSAC, 1980

12. A. J. Albrecht and J. E. Gaffney, Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," IEEE Transactions on Software Engineering, November 1983

13. Computer Sciences Corporation, Informational Memorandum, "Extracting Specification Measures From Flight Dynamics Software Requirements," W. Agresti, December 1984

14. --, Informational Memorandum, "Case Study in Recasting Flight Dynamics Software Requirements Using the Composite Specification Model (CSM)," W. Agresti, December 1984

15. W. W. Agresti, "Measuring Program Maintainability," Journal of Systems Management, vol. 33, no. 3, 1982

16. Index Technology Corporation, Excelerator Reference Guide, Release 1.11, 5 Cambridge Center, Cambridge, Massachusetts, 02142, 1984

17. Software Engineering Laboratory, SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. Scheffer and T. Velez, November 1978

BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-ORIGINATED DOCUMENTS

SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976

SEL-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu and D. S. Wilson, September 1977

SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977

SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

[1]SEL-78-002, FORTRAN Static Source Code Analyzer (SAP) User's Guide, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

SEL-78-102, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1), W. J. Decker and W. A. Taylor, September 1982

SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, D. S. Wilson and B. Chu, September 1978

SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978

SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer and C. E. Velez, November 1978

SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978

SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979

SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979

SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979

SEL-80-001, Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, A. L. Green, and C. E. Goorevich, February 1980

SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker and C. E. Goorevich, May 1980

SEL-80-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, T. Welden, M. McClellan, and P. Liebertz, May 1980

[1]SEL-80-004, System Description and User's Guide for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, W. J. Decker, J. G. Garrahan, et al., October 1980

SEL-80-104, Configuration Analysis Tool (CAT) System Description and User's Guide (Revision 1), W. Decker and W. Taylor, December 1982

SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980

SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980

SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

[1]SEL-81-001, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., September 1981

SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982

[1]SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. C. Wyckoff, G. Page, and F. E. McGarry, September 1981

SEL-81-102, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide Revision 1, P. Lo and D. Wyckoff, July 1983

[1]SEL-81-003, Data Base Maintenance System (DBAM) User's Guide and System Description, D. N. Card, D. C. Wyckoff, and G. Page, September 1981

[1]SEL-81-103, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, P. Lo and D. Card, July 1983

SEL-81-203, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, P. Lo, June 1984

[1]SEL-81-004, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., September 1981

SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

[1]SEL-81-005, Standard Approach to Software Development, V. E. Church, F. E. McGarry, G. Page, et al., September 1981

[1]SEL-81-105, Recommended Approach to Software Development, S. Eslinger, F. E. McGarry, and G. Page, May 1982

SEL-81-205, Recommended Approach to Software Development, F. E. McGarry, G. Page, S. Eslinger, et al., April 1983

SEL-81-006, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide, W. Taylor and W. J. Decker, December 1981

[1]SEL-81-007, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, A. L. Green, et al., February 1981

SEL-81-107, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, W. A. Taylor, and E. J. Smith, February 1982

SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker and F. E. McGarry, March 1981

[1]SEL-81-010, Performance and Evaluation of an Independent Software Verification and Integration Process, G. Page and F. E. McGarry, May 1981

SEL-81-110, Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics, G. Page and F. McGarry, December 1983

SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

SEL-81-012, The Rayleigh Curve As a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981

SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

SEL-82-001, Evaluation of Management Measures of Software Development, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-002, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. A. Taylor and W. J. Decker, August 1982

SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo, September 1982

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

[1]SEL-82-005, Glossary of Software Engineering Laboratory Terms, M. G. Rohleder, December 1982

SEL-82-105, Glossary of Software Engineering Laboratory Terms, T. A. Babst, F. E. McGarry, and M. G. Rohleder, October 1983

[1]SEL-82-006, Annotated Bibliography of Software Engineering Laboratory (SEL) Literature, D. N. Card, November 1982

[1]SEL-82-106, Annotated Bibliography of Software Engineering Laboratory Literature, D. N. Card, T. A. Babst, and F. E. McGarry, November 1983

SEL-82-206, Annotated Bibliography of Software Engineering Laboratory Literature, D. N. Card, Q. L. Jordan, and F. E. McGarry, November 1984

SEL-82-007, Proceedings From the Seventh Annual Software Engineering Workshop, December 1982

SEL-82-008, Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory, V. R. Basili and D. M. Weiss, December 1982

SEL-83-001, An Approach to Software Cost Estimation, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

SEL-83-002, Measures and Metrics for Software Development, D. N. Card, F. E. McGarry, G. Page, et al., March 1984

SEL-83-003, Collected Software Engineering Papers: Volume II, November 1983

[1]SEL-83-004, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) User's Guide, T. A. Babst and W. J. Decker, November 1983

SEL-83-104, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) User's Guide, T. A. Babst, W. J. Decker, P. Lo, and W. Miller, August 1984

[1]SEL-83-005, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) System Description, P. Lo and W. J. Decker, November 1983

SEL-83-105, Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) System Description, P. Lo, W. J. Decker, and W. Miller, August 1984

SEL-83-006, Monitoring Software Development Through Dynamic Variables, C. W. Doerflinger, November 1983

SEL-83-007, Proceedings From the Eighth Annual Software Engineering Workshop, November 1983

SEL-84-001, Manager's Handbook for Software Development, W. W. Agresti, V. E. Church, and F. E. McGarry, April 1984

SEL-84-002, Configuration Management and Control: Policies and Procedures, Q. L. Jordan and E. Edwards, December 1984

SEL-84-003, Investigation of Specification Measures for the Software Engineering Laboratory (SEL), W. Agresti, V. Church, and F. E. McGarry, December 1984

SEL-RELATED LITERATURE

Agresti, W. W., Definition of Specification Measures for the Software Engineering Laboratory, Computer Sciences Corporation, CSC/TM-84/6085, June 1984

[2]Agresti, W. W., F. E. McGarry, D. N. Card, et al., "Measuring Software Technology," Program Transformation and Programming Environments. New York: Springer-Verlag, 1984

[3]Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Banks, F. K., "Configuration Analysis Tool (CAT) Design," Computer Sciences Corporation, Technical Memorandum, March 1980

[3]Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1979

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

[3]Basili, V. R., and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

[3]Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

[2]Basili, V. R., and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," Communications of the ACM, January 1984, vol. 27, no. 1

[3]Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981

[2]Basili, V. R., R. W. Selby, and T. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects," IEEE Transactions on Software Engineering, November 1983

Basili, V. R., and J. Ramsey, Structural Coverage of Functional Testing, University of Maryland, Technical Report TR-1442, September 1984

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

[2]Basili, V.R., and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, University of Maryland, Technical Report TR-1235, December 1982

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

[3]Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

[3]Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

[3]Basili, V. R., and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

[2]Card, D. N., "Early Estimation of Resource Expenditures and Program Size," Computer Sciences Corporation, Technical Memorandum, June 1982

[2]Card, D. N., "Comparison of Regression Modeling Techniques for Resource Estimation," Computer Sciences Corporation, Technical Memorandum, November 1982

Card, D. N., and V. E. Church, "Analysis Software Requirements for the Data Retrieval System," Computer Sciences Corporation Technical Memorandum, March 1983

Card, D. N., V. E. Church, W. W. Agresti, and Q. L. Jordan, "A Software Engineering View of Flight Dynamics Analysis System," Parts I and II, Computer Sciences Corporation Technical Memorandum, February 1984

Card, D. N., Q. L. Jordan, and V. E. Church, "Characteristics of FORTRAN Modules," Computer Sciences Corporation Technical Memorandum, June 1984

[3]Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

[2]Doerflinger, C. W., and V. R. Basili, "Monitoring Software Development Through Dynamic Variables," Proceedings of the Seventh International Computer Software and Applications Conference. New York: Computer Societies Press, 1983

Freburger, K., "A Model of the Software Life Cycle" (paper prepared for the University of Maryland, December 1978)

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Hislop, G., "Some Tests of Halstead Measures" (paper prepared for the University of Maryland, December 1978)

Lange, S. F., "A Child's Garden of Complexity Measures" (paper prepared for the University of Maryland, December 1978)

McGarry, F. E., E. C. Edwards, K. Liu, and G. Page, Software Conversion History: Flight Dynamics System, June 1984

McGarry, F. E., G. Page, and R. D. Werking, Software Development History of the Dynamics Explorer (DE) Attitude Ground Support System (AGSS), June 1983

Miller, A. M., "A Survey of Several Reliability Models" (paper prepared for the University of Maryland, December 1978)

National Aeronautics and Space Administration (NASA), NASA Software Research Technology Workshop (proceedings), March 1980

Page, G., "Software Engineering Course Evaluation," Computer Sciences Corporation, Technical Memorandum, December 1977

Page, G., F. E. McGarry, and D. N. Card, "A Practical Experience With Independent Verification and Validation," Proceedings of the Eighth International Computer Software and Applications Conference, November 1984

Parr, F., and D. Weiss, "Concepts Used in the Change Report Form," NASA, Goddard Space Flight Center, Technical Memorandum, May 1978

Reiter, R. W., "The Nature, Organization, Measurement, and Management of Software Complexity" (paper prepared for the University of Maryland, December 1976)

Scheffer, P. A., and C. E. Velez, "GSFC NAVPAK Design Higher Order Languages Study: Addendum," Martin Marietta Corporation, Technical Memorandum, September 1977

Turner, C., and G. Caron, A Comparison of RADC and NASA/SEL Software Development Data, Data and Analysis Center for Software, Special Publication, May 1981

Turner, C., G. Caron, and G. Brement, NASA/SEL Data Compendium, Data and Analysis Center for Software, Special Publication, April 1981

Weiss, D. M., "Error and Change Analysis," Naval Research Laboratory, Technical Memorandum, December 1977

Williamson, I. M., "Resource Model Testing and Information," Naval Research Laboratory, Technical Memorandum, July 1979

[3]Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

[2]Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," Empirical Foundations for Computer and Information Science (proceedings), November 1982

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977

Zelkowitz, M. V., and J. Sukri, "Evaluation of the FDAS Prototype as a Software Development System" (paper prepared for the University of Maryland, February 1984)

---

[1]This document superseded by revised document.

[2]This article also appears in SEL-83-003, Collected Software Engineering Papers: Volume II, November 1983.

[3]This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume I, July 1982.